Automated Photo to Watercolor Painting with realistic wet-in-wet

Michael Fotheringham
JixiPix Software, mikef@jixipix.com

Kristal Fotheringham
JixiPix Software, kristal@jixipix.com

***Preface* – This technique is the basis of a product launched by JixiPix Software called 'Watercolor Studio' that shipped in August 2017. This algorithm is based on the 2nd version that was launched the end of December, 2017. This algorithm has since been revised and rewritten multiple times but the basis is still the same.**

Introduction

There have been numerous papers and designs through the years on converting a photo into a realistic watercolor painting. Most emphasized filter based approaches which has advantages and disadvantages. The biggest disadvantage is that a filter based approach has no real 'smarts' and it makes it very difficult to simulate how a real artist would paint. Our system breaks up an image into regions and renders each region individually. This gives us control of negative space and overlap, scalability to very large images, and having the ability of each region to check its neighbors size, color, luminosity, etc.

New Approach

Our system approaches the problem in a direct approach as a partial filter based technique and region based technique. We first adjust the image color to be better in-line with a real watercolor painting by increasing saturation and possibly overlaying color texture that resembles real watercolor painting styles. We then abstract the image and convert to regions. Next, we add hand tremor and wobbling, render regions and flow pigment across scene, and finally add in wet edges, paper texture and granulation.

Image Preparation

Watercolor paintings are typically quite a bit more saturated than a regular photo. They can also be painted with certain tones or mood. Image preparation entails saturation, contrast, color shifts, LUT, etc. to set up the 'mood' for the painting. A simple procedure would be to just increase the saturation without blowing out or clipping the colors.

Image Abstraction

Watercolor paintings are typically not overly detailed and abstracting the image will cut down on the detail while also aiding in creating areas or regions. Previous authors like Bousseau, et.al[1], and Doran, et.al[2] presented various ways to abstract the image. The choice in this is less critical to the overall look of watercolor and various factors can play into the choice on which segmentation/abstraction is used. David Stutz [3] has a thorough write up on the current state (circa 2017) on superpixels, while Felzenszwalb, et.al [4] details a quick graph-based image segmentation approach. Figure 1

Figure 1. Abstracted Images

Hand Tremor and Wobbling

Hand tremor and wobbling is now added to each region. This gives the end result of negative space and overlaps between areas as well as making the edges of each region appear to flow into the paper texture. When an artist paints they rarely will have completely straight lines and this process will give more of a natural look to the abstracted regions. The first stage will add the hand tremor with the wobbling applied after.

For Hand Tremor we take a simple offset approach by generating multiple different Perlin noise textures[1] at a larger size. We then process each region by taking the

normal to each control point and extending or contracting based on the pre-generated noise texture.

K = texture value
N = surface normal
C = Control Point
A = max tremor distance
$C' = C*N*(K - 0.5)*A$

We then do a similar process and change out the Perlin noise for a scanned paper texture that has fine detail. Texture value will vary in a high frequency pattern to give less 'smooth' areas and more jitter noise. A larger A will give hand tremor more negative space and overlap while also giving a more abstract style to the regions.

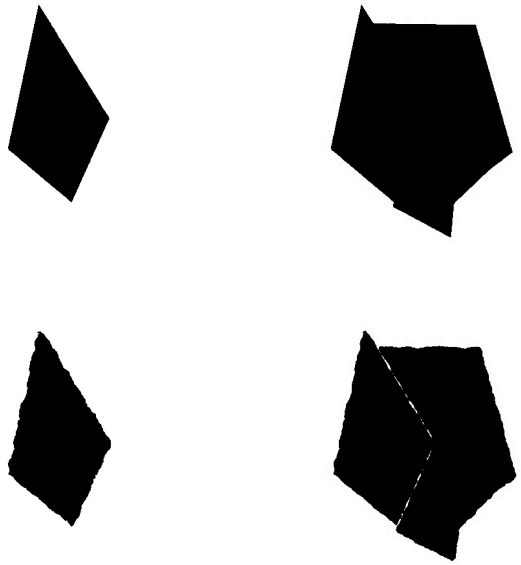


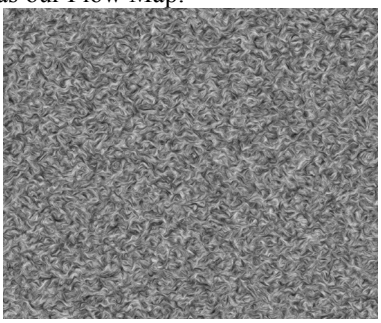
Figure 2. Examples of regions before and after hand tremor and wobble added. Top-left is region 1 and top-right is both region 1 and region 2. Bottom-left is region 1 rendered with hand tremor and wobble. Bottom-right shows both regions combined

PERCEIVED COLOR

During the rendering phase we will use a perceived color algorithm to determine color ordering. This has emphasis on perceived color rather than straight gray conversion and we can adjust our function on the fly based on image type or user parameters. By default, the equation is simply $P_c = 1 - (r*.241 + g*.691 + b*.068)$. Values with a darker value (higher P_c) will be rendered last and have more emphasis in the final painting.

FLOW CONTROL

Ren-Jie Wang, et.al [6] proposed a line integral convolution (LIC) vector field and Adaptive Length Line Integral Convolution for smooth pigment flowing. We stay with LIC with an anisotropic diffusion applied for smoothing purposes and use this as our Flow Map.



Example of rendered flow map.

The addition to the flow is a smart region technique where the ink flows but only where the hue and intensity change is small between the current rendering region and the neighbor region [9]. This keeps ink from flowing freely into surrounding areas with large changes. A shade of blue will freely flow paint into a close secondary region with a shade of blue. This gives the perception of wet-in-wet with close regions freely flowing into neighboring regions (and possibly through multiple regions). Non-salient regions and regions with a lower P_c are given a larger flow than salient regions and higher P_c. This gives the perception of lighter colors and non-salient regions flowing more and creating more wet-in-wet pigment flow.

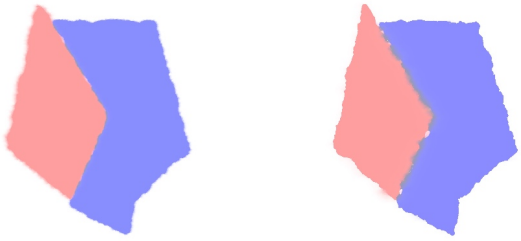


Figure 3. Hand Tremor and wobble added to regions painted with region color and perceived color opacity. Top-left is flow in raw form. Top-right is flow with region aware neighbor.

RENDERING PHASE

The rendering phase consists of rendering each region into the scene. First step is to diffuse the region using the Flow Map. Rendering order is determined by a [5][9][6] Saliency Distance Field and perceived light to dark colors. Non-salient regions are rendered first and salient regions composited on top with both sets rendered using P_c. We also assign an opacity equal to P_c for each region. This will give the perception of semi-transparent pigment while retaining darker/salient regions to be less transparent.

Algorithm

```
for (all light region to dark region in non-salient areas) {
        paint region into flow layer
        FlowPigment()
        render pigment to scene using 'darken' color mode
}
for (all light region to dark region in salient areas) {
        paint region into flow layer
        FlowPigment()
        render pigment to scene using 'darken' color mode
}
```

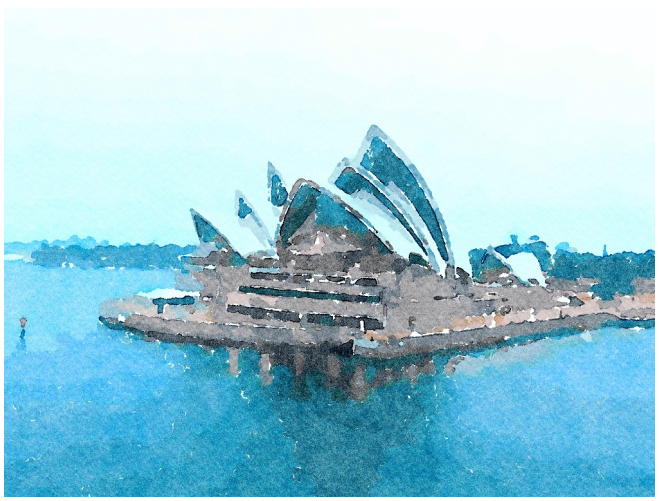
OTHER WATERCOLOR EFFECTS

Wet edges can be created using similar techniques to above hand tremor and wobbling phase followed by the rendering phase without Flow Control. We use a different perlin noise and render only *N* pixels from the inset edge of the region. This gives an outline region boundary roughened different than the rendered region. Where both regions are active we darken scene with the regions color. This achieves a wet edge technique with inconsistencies in each region.

Granulation, low-frequency turbulent flow and paper texture can be added using techniques previously described [1][6][7][8][9].

FINAL SAMPLES

All the samples below include the added wet edge, granulation, low-frequency turbulent noise and paper texture.



CONCLUSION AND THOUGHTS

Our technique is very simple but produces a very realistic and pleasing result. Keeping the areas of the abstracted photo as regions gives us the ability to scale this technique to large size photos while still retaining quality. Pixel/filter based solutions would emphasize noise and become jaggy as you scaled larger. Regions also give us total control (or users total control) of the tremor, wobbling, and negative space.

FUTURE IDEAS

As stated in the preface, this algorithm is based on our v2 of our watercolor rendering effect. V3 added in smarter wet edges that use a particle system to propagate the pigment to the edges of each region. V3.5 added in smarter low-frequency turbulent noise that was unique for each region. A new version (unnamed) will be adding in deep learning and replacing the salient areas with partial object recognition and masking of objects as well as switching the region LIC flow system to a particle system that dynamically handles paper texture changes and paper tilt.

REFERENCES

[1] Interactive Watercolor rendering with temporal coherence and abstraction – Adrien Bousseau, Matt Kaplan, Joell Thollot, Francois X. Sillion – 2006

[2] Expressive Rendering with Watercolor – Patrick J. Doran, John Hughes – 2010

[3] Superpixel Benchmark "http://davidstutz.de/projects/superpixel-benchmark/"– David Stutz – 2016

[4] Efficient Graph-Based Image Segmentation – Pedro F. Felzenszwalb, Daniel P. Huttenlocher - 2004

[5] Global Contrast based Salient Region Detection – Ming-Ming Cheng, Guo-Xin Zhang, Niloy J. Mitra, Xiaolei Huang, Shi-Min Hu – 2011

[6] Effective Wet-in-Wet Flow Synthesis for Ink Diffusion – Ren-Jie Want, Chung-Ming Wang – 2010

[7] E. Lei and C.-F. Chang, "Real-time rendering of watercolor effects for virtual environments," in *Proceedings of the PCM '04*, 2004, pp. 474–481.

[8] T. Luft and O. Deussen, "Interactive watercolor animations," in *Poster of the PG '05*, 2005, pp. 7–9.

[9] Towards Photo Watercolorization with Artistic Verisimiltude – Miaoyi Wang, Bin Wang, Yun Fei, Kanglai Qian, Wenping Wang, Jiating Chen, and Jun-Hai Yong - 2014

AUTHOR INFORMATION

Michael Fotheringham is currently an effect engineer for JixiPix Software. He has a long history of doing special effect work dating back to 1987 and has worked for many pioneering effect and 3D software companies.

Kristal Fotheringham is currently an artist for JixiPix Software. She has a long history in the art field including work from everything from advertising agencies to fine arts, to interior design. Kristal is not directly an effect engineer but works closely with the engineering team when designing and implementing algorithms.